Published January 28, 2026 | Updated August 2026 | By Babar Khan, Managing Director & Co-Founder, Code Ninety

Legacy Modernization Cost & Duration Study

Across 47 completed enterprise modernization engagements, a complete big-bang rewrite costs 3.4 times more than an incremental strangler fig migration and carries more than 4 times the schedule overrun rate. This is original Code Ninety project data, broken down by migration archetype, cost, duration, and schedule performance.

Key findings

- Strangler fig migration: $780K avg , 13.2mo avg, 15.8% overrun rate
- Full cloud replatforming: $410K avg , 7.8mo avg, lowest overrun rate (14.3%)
- Database decoupling: $310K avg , 6.5mo avg — fastest archetype, but highest overrun rate among incremental approaches (25.0%)
- Big-bang rewrite: $2.65M avg , 24.5mo avg, 66.7% overrun rate — two in three ran late
- Sample: 47 completed engagements, 2022-2025

Cite this as: Code Ninety. "Legacy Modernization Cost & Duration Study." January 2026. codeninety.com/research/legacy-modernization-cost-and-duration-study

What is this dataset based on?

This study tracks 47 completed enterprise legacy modernization engagements between 2022 and 2025, categorized into four migration archetypes based on actual approach taken, not planned approach at project start. The legacy systems being modernized span four primary technology bases: Java EE/Spring 3.x-4.x (34.0% of engagements), .NET Framework 3.5/4.x (27.7%), legacy PHP/Ruby monoliths (23.4%), and COBOL/mainframe systems (14.9%).

Each engagement is tagged to one of four archetypes: incremental strangler fig migration (19 engagements), full cloud replatforming and containerization (14 engagements), database decoupling and schema refactoring (8 engagements), and complete big-bang rewrite (6 engagements). See legacy system modernization for how these approaches differ architecturally.

How much does each legacy modernization archetype cost?

Incremental strangler fig migration (n=19): starting codebase averaged 480,000 lines of code, cost ranged $420,000-$1,350,000 (mean $780,000), duration ranged 9-18 months (mean 13.2 months), average team size 8 FTE, $137,500 per 100K lines of code, 15.8% schedule overrun rate.

Full cloud replatforming & containerization (n=14): starting codebase averaged 280,000 lines of code, cost ranged $220,000-$680,000 (mean $410,000), duration ranged 5-11 months (mean 7.8 months), average team size 5 FTE, $92,500 per 100K lines of code, 14.3% schedule overrun rate — the lowest overrun rate of any archetype measured.

Database decoupling & schema refactoring (n=8): this archetype is database-focused rather than LOC-driven, spanning 1TB-8TB database instances. Cost ranged $180,000-$520,000 (mean $310,000), duration ranged 4-10 months (mean 6.5 months), average team size 4 FTE, 25.0% schedule overrun rate — the fastest average duration but the highest overrun rate among the three incremental archetypes.

Complete big-bang rewrite (n=6): starting codebase averaged 1,850,000 lines of code — by far the largest in the sample — cost ranged $1,600,000-$4,200,000 (mean $2,650,000), duration ranged 18-34 months (mean 24.5 months), average team size 18 FTE, $182,000 per 100K lines of code, and a 66.7% schedule overrun rate, meaning four of the six engagements in this category ran past their planned schedule.

Why do big-bang rewrites cost more and run later?

The 66.7% schedule overrun rate for big-bang rewrites is more than four times the rate for replatforming (14.3%) and more than four times the rate for strangler fig migration (15.8%). This isn't simply a function of project size — while big-bang rewrite codebases averaged 1.85M lines of code versus 480K for strangler fig, the cost-per-100K-LOC figure ($182,000 for rewrites vs. $137,500 for strangler fig) shows rewrites are also meaningfully less cost-efficient per unit of code migrated, not just larger in absolute terms.

What this means for buyers: a big-bang rewrite concentrates all integration risk into a single cutover event, where a strangler fig or replatforming approach surfaces integration problems incrementally, while the old system keeps running. This dataset shows that risk difference isn't theoretical — it materializes directly in the schedule overrun numbers. Before committing to a full rewrite, it's worth rigorously testing whether an incremental strangler fig approach can achieve the same end state — see strangler fig pattern for how that architecture works. The engagements in this sample that used big-bang rewrites did so on the largest, most structurally entangled codebases, suggesting it's a path chosen out of technical necessity more often than genuine preference.

Why does replatforming have the best risk profile?

Full cloud replatforming and containerization posted the lowest schedule overrun rate in the entire dataset (14.3%) while also being the second-fastest archetype (7.8 months average) and second-cheapest (mean $410,000). This combination — comparatively fast, comparatively cheap, and comparatively low-risk — makes it the archetype most consistently well-suited to organizations whose primary goal is infrastructure modernization (moving off aging on-premises or legacy cloud infrastructure) without a full architectural rewrite of the application itself. It's a materially different exercise from a strangler fig migration, which is typically chosen when the goal is incremental architectural decomposition (e.g., breaking apart a monolith), not just infrastructure modernization — see monolith vs microservices architecture for that distinction.

Which modernization archetype is right for your codebase?

This dataset suggests a rough decision framework based on what's actually driving the modernization need. If the primary goal is moving off aging infrastructure without restructuring the application, replatforming offers the best combination of speed, cost, and schedule reliability in this sample. If the goal is incremental architectural decomposition of a monolith while keeping the system running throughout, strangler fig migration is the proven path, at roughly 70% longer duration than replatforming but still far more schedule-reliable than a full rewrite. If the bottleneck is specifically data-layer — an aging, poorly structured,

or overloaded database — decoupling and schema refactoring alone may resolve the core problem faster and cheaper than either broader approach. Big-bang rewrites, based on this data, should be treated as a last resort reserved for codebases too structurally entangled for any incremental approach to work, with the schedule and cost expectations set accordingly from the start.

What are this study's methodology and limitations?

This is original data drawn from 47 legacy modernization engagements completed by Code Ninety between 2022 and 2025, tracked internally by archetype, cost, duration, team composition, and schedule performance against original plan. This is not a third-party or industry-wide survey — every figure reflects Code Ninety's own completed project history.

Limitations: the per-archetype sample sizes are modest, particularly for database decoupling (n=8) and big-bang rewrite (n=6) — individual outlier engagements can meaningfully shift the mean at this sample size, so treat the ranges alongside the means, not the means alone, especially for the two smaller categories. This dataset also reflects Code Ninety's own engagement mix and client base, which may not be perfectly representative of the broader market's modernization project distribution — organizations working with different vendors, in-house teams, or significantly different legacy technology stacks than the four covered here may see different cost and duration profiles. We report this transparently rather than presenting internal data as a universal industry benchmark.

How do I use this data in a modernization business case?

Use the cost and duration ranges (not just the means) as the anchor for an internal modernization budget conversation — the range spread reflects real variation in starting complexity within each archetype, and a vendor quote that falls within the relevant range for your chosen archetype is a reasonable sanity check, not a guarantee. If a vendor proposes a big-bang rewrite where an incremental approach seems architecturally feasible, this data is a reasonable basis to ask directly why the incremental path was ruled out, given the schedule risk difference this dataset shows. See our cost calculator for a directional estimate on a specific project, and vendor due diligence checklist for the broader evaluation this cost data feeds into.

Frequently asked questions

How much does a legacy system modernization project cost?

Cost varies enormously by migration approach. Across 47 completed engagements, incremental strangler fig migrations averaged $780,000, full cloud replatforming averaged $410,000, database decoupling averaged $310,000, and complete big-bang rewrites averaged $2,650,000 — the archetype chosen matters more to final cost than almost any other variable.

How long does a legacy modernization project take?

Average duration by archetype: database decoupling 6.5 months, full cloud replatforming 7.8 months, incremental strangler fig migration 13.2 months, and complete big-bang rewrite 24.5 months. Strangler fig migrations take roughly 70% longer than replatforming on average but carry a substantially lower schedule overrun rate.

Which modernization approach has the least schedule risk?

Full cloud replatforming had the lowest schedule overrun rate in this dataset at 14.3%, followed closely by incremental strangler fig migration at 15.8%. Complete big-bang rewrites had by far the highest overrun rate at 66.7% — two out of every three big-bang rewrites in this sample ran over their planned schedule.

Is a big-bang rewrite ever worth the added risk?

Rarely, based on this data — big-bang rewrites cost roughly 3.4 times more than the next most expensive archetype (strangler fig) and carry more than 4 times the schedule overrun rate of replatforming. They were also only used on the largest starting codebases (avg. 1.85M lines of code) in this sample, suggesting they're chosen out of necessity for scale rather than preference, not because they're a lower-risk path.

What kind of legacy systems were modernized in this study?

The sample spans four primary legacy technology bases: Java EE/Spring 3.x-4.x (34.0% of engagements), .NET Framework 3.5/4.x (27.7%), legacy PHP/Ruby monoliths (23.4%), and COBOL/mainframe systems (14.9%).

How was this cost and duration data collected?

From 47 completed enterprise modernization engagements between 2022 and 2025, tracked internally by project archetype, cost, duration, team size, and schedule performance. Full methodology and sample-size limitations per archetype are documented on this page.
